Dimension Reduction and Clustering of High Dimensional Data using Auto-Associative Neural Networks

Zalhan Mohd Zin
UniKL MFI
Bandar BaruBangi
Malaysia

Rubiyah Yusof
Universiti Teknologi Malaysia
UTM IC Kuala Lumpur
Malaysia

Ehsan Mesbahi
Newcastle University
Newcastle Upon Tyne UK

ABSTRACT

The task to capture and interpret information hidden inside high-dimensional data can be considered very complicated and challenging. Usually, dimension reduction technique may be considered as the first step to data analysis and exploration. The focus of this paper is on high-dimensional data dimension reduction using a supervised artificial neural networks technique known as Auto-Associative Neural Networks (AANN). The AANN can be considered as a powerful tool in data analysis and clustering with the ability to deal with linear and nonlinear correlation among variables. This technique is sometimes referred to as nonlinear principal component analysis (NLPCA), Encoding-Decoding networks, or bottleneck neural networks (BNN) due to its unique structure. It reduces high-dimensional data into low-dimensional data on its bottleneck layer which can later be used for data transmission, clustering and visualization. In this paper, a structurally flexible AANN is developed by using high level computer language, applied and studied on two case studies of Iris flowers and Italian olive oils datasets. The purpose of the work was to investigate the ability of AANN to reduce dimension of high-dimensional data on small (Iris) and large (Olive) datasets. The results have shown that AANN has been able to compress high-dimensional data into only one or two non-linear principal components at its bottleneck layer with the highest accuracy of 98.9% and 82.1% for both datasets respectively. AANN has also managed to perform accurately in both reducing dimension and clustering data by only using small portion of training dataset.

General Terms

Artificial Intelligence, Artificial Neural Networks, Data Clustering

Keywords

Dimension Reduction, Auto-Associative Neural Networks

1. INTRODUCTION

Understanding and interpreting high-dimensional data is usually seen as very challenging and complicated. Data compression, dimension reduction and the visualization of these high-dimensional data could actually provide a mathematically simpler way to have a better understanding and meaningful interpretation of characteristics of a complex data set. In [1] [2], it was mentioned that dimension reduction has been considered as the first step to be taken in order to understand the significant patterns inside high-dimensional data. In fact, there are many techniques of dimension reduction that have been used previously. Principal Component Analysis (PCA) is commonly used to find pattern of high-dimensional data as described in [3]. This technique has been described, used, compared and evaluated in many research works and publications such as in [1] [2] [4] [5] [6]. However, the effectiveness of linear PCA technique is quite limited because it cannot deal efficiently with nonlinear correlated variables. For nonlinear data set problems, nonlinear dimension reduction or projection methods are generally applied. Some examples of these techniques are Multidimensional Scaling (MDS), Locally Linear Embedding (LLE), Isomap, Kernel PCA [7], Self-Organizing Maps (SOM) [8] [9] [10] and also Auto-Associative Neural Networks (AANN) [1] [2]. The latter, AANN or also known as Bottleneck Neural Networks (BNN) has been previously used for data compression or dimension reduction particularly in the field of information retrieval, chemical applications, missing data estimations and image compressions [2] [6] [11] [12]. But, there was still very little attention given to AANN. In [6], the authors mentioned that only 232 English language works related to AANN were cited unlike the more generic topic of artificial neural network which was cited 73,397 times in the last decade. More specifically in their study on literatures, AANN has most frequently been applied in the area of information retrieval and storage (19%). Besides that, it has also been applied in pattern classifier and recognizer (18%), image recognition and image processing (13%), anomaly and fault detection (12%), system modeling (financial modeling) (12%), dimensionality reduction (8%), filtering (6%) and pure research where the researchers are pushing the limits of the architecture and experimenting with potentially useful variants. This study has indicated that there could be many potential applications to which the AANN can be applied and studied. However, this technique has not yet been largely studied or exploited in dealing with the challenges from high dimensional data dimension reduction and clustering.

One of the challenges in dimension reduction task is on how to retain as much information as possible of high-dimensional data. More specifically for AANN it is important to know that much information it can retain within its inherent nonlinear structure (bottleneck layer) after the network is successfully trained. In this research, the data compression, clustering and visualization abilities of AANN have been and further investigated. The AANN mathematical model has been developed using high level computer language and applied on Iris flowers and Italian olive oils datasets. The algorithm has been developed using C/C++ language and a series of experimental works conducted using small and large sizes of training and testing datasets.

2. METHODS

2.1 Auto-Associative Neural Networks

The AANN has often been regarded as an alternative to PCA for unsupervised learning, clustering, and outlier detection [13]. It can also be known as Non Linear PCA (NLPCA) [1] or Bottleneck Neural Networks (BNN) [2]. AANN can be applied to the same problems as in conventional PCA such as dimension reduction and visualization, fault detection etc. AANN becomes more interesting as machine learning algorithm because it can deal with non-linear data problems more efficiently than PCA [2].

AANN is a feed forward network and is trained to map its input vectors back to the same input vector. In other words, AANN's output layer is identical to its input layer. As described in [14], AANN is considered as a particular class of neural networks and can be described as circuits of highly interconnected units with adjustable interconnection weights. A conventional Artificial Neural Network (ANN) is structured to imitate human brain abilities of learning and memorizing. This technique uses similar concept with our biological neural system where a neuron is used as the smallest functional unit. The neuron receives simultaneously a number of input signals *n*. Each of these signals is associated with their respective weights, *w*. The weighted sum of signals becomes the argument of the activation function. The value obtained from this function becomes the output of the neuron. The activation function can be linear or non-linear function. This allows neuron to perform linear or nonlinear calculation operations. An illustration of the functional unit or neuron is shown in Figure 1.

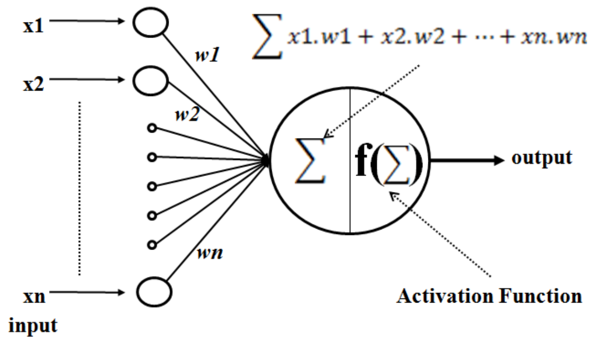

Figure 1: Functional unit or neuron of the network. $x_i$is n-dimensional input and w_i is an n-dimensional weight with $i = \{1,..,n\}$.

The structure of AANN usually consists of several layers: input layer, map layer, bottleneck layer, de-map layer and output layer as shown in Figure 2. The map, bottleneck and de-map layers are its hidden layers. The input layer and output layer are the same, denoted with X. This structure could also be seen as a combination of two different networks: compression network and de-compression network, as described in [1] [2] [14]. Both of these networks "meet" at bottleneck layer located in the middle of AANN structure. In compression network, the input is known but the output is not known while the opposite situation occurs in de-compression network where the input is not known but the output is known. With this combination of networks, the output of compression network becomes the input of de-compression network.

During the training of AANN, the high-dimensional data is firstly compressed to few potential variables at bottleneck layer in compression network. These variables correspond to the number of neurons or nodes in the bottleneck. This number must be smaller than the number of nodes at input or output layers. Then later, the output of the bottleneck is decompressed throughout the decompression network. The AANN training continues to map input vectors to their corresponding output vectors. Each value at input nodes is adjusted to be as identical as possible to its respective value at output nodes. Once the training is completed, the bottleneck nodes of AANN represent a type of nonlinear principal components, which are frequently more relevant than PCA for analyzing nonlinear or real-world datasets.

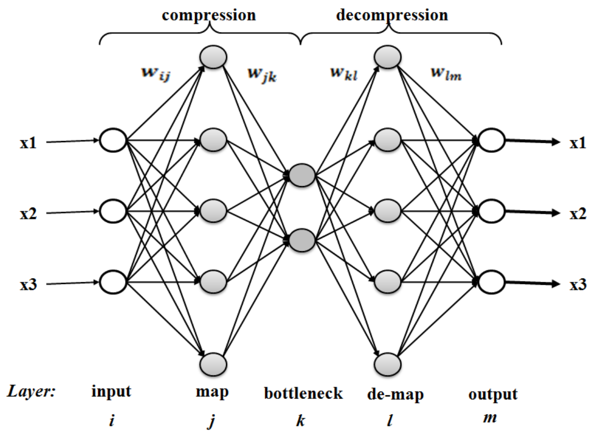

Figure 2: Structure of AANN that consists of 3 nodes at input and output layers, 5 nodes at map and de-map layers and 2 nodes at bottleneck layer.

The ability of AANN to deal with linear and nonlinear correlation between variables is directly related to the type of the transfer function (also known as activation functions) used in each neuron in its layers. According to [1] [14], the neurons in map and de-map layers must use non-linear or sigmoid transfer functions. This is necessary to ensure proper functioning of AANN in dealing with non-linear problems [2]. Meanwhile, the neurons in bottleneck and output layers could be either linear transfer functions or sigmoid transfer functions. In our case, the sigmoid function (continuous and monotonically increasing) has been used in all layers and neurons. This function takes input values X and scales them into the range between 0 and 1. It is defined as follows:

$$\sigma(X) = \frac{1}{1+e^{-X}} \qquad \text{(eq-1)}$$

The learning process of AANN starts with random initialization of all weights *w*, which link all neurons in all layers in the network (Figure 2). All input data values, X, are normalized to {0,..,1}. As a feed-forward network, the algorithm calculates the values at all nodes at map layer using the functional unit as shown in Figure 1. Considering $M_{(i,j)}$ function for this operation is defined as:

$$M_{(i,j)} = \sigma(\sum_{j=1}^{nbmp} \sum_{i=1}^{nbinput} x_i w_{ij}) \qquad \text{(eq-2)}$$

with *nbmp* and *nbinput* are number of nodes in map and input layers respectively. This feed forward processes continue throughout bottleneck, de-map and output layers using the following equations:

$$B_{(i,j,k)} = \sigma(\sum_{k=1}^{nbtl} \sum_{j=1}^{nbmp} M_{(i,j)} w_{jk}) \qquad \text{(eq-3)}$$

$$D_{(i,j,k,l)} = \sigma(\sum_{l=1}^{nbdmp} \sum_{k=1}^{nbtl} B_{(i,j,k)} w_{kl}) \qquad \text{(eq-4)}$$

$$O_{(i,j,k,l,m)} = \sigma(\sum_{m=1}^{nboutput} \sum_{l=1}^{nbdmp} D_{(i,j,k,l)} w_{lm}) \qquad \text{(eq-5)}$$

with *nbtl, nbdmp* and *nboutput* represent the number of nodes and $B_{(i,j,k)}$, $D_{(i,j,k,l)}$, and $O_{(i,j,k,l,m)}$ represent functions at bottleneck, de-map and output layers respectively. So, at iteration p, the value obtained at a certain neuron m, $x_m(p)$ at output layer is equaled to $O_{(i,j,k,l,m)}(p)$. The input signals are actually propagated through network from left to right (input layer to output layer). On the other hand, back-propagation technique as described in [15] has been used for all weights adjustment. Once the signals arrive at all nodes m in output layer, the error signals are then calculated using the following equation:

$$e_m = x_{d,m}(p) - x_m(p) \qquad \text{(eq-6)}$$

where $x_{d,m}(p)$ is the desired output and $x_m(p)$ is the actual output of node m at iteration p. These errors are then back-propagated from output layer back to input layer in order to update the network's weights. The rule used to update weights between output layer and de-map layer is:

$$w_{lm}(p+1) = w_{lm}(p) + \Delta w_{lm}(p) \qquad \text{(eq-7)}$$

where l represent the nodes at de-map layer and $\Delta w_{lm}(p) = \alpha \times y_l(p) \times \delta_m(p)$ is weight correction where $\delta_m(p)$ is the gradient error at neuron m in the output layer at iteration p and α is the learning rate. When all weights w_{ij} are updated through back-propagation error signals, iteration p is considered complete. The Mean Square Error (MSE) between all the nodes i at input layer and all the nodes m at output layer is now calculated as follows:

$$MSE = \frac{1}{N} \sum_{i=1;m=1}^{nbinput\ ;nboutput} (X_i - X_m)^2 \qquad \text{(eq-8)}$$

where N represents number of input and output dimensions. As mentioned previously, AANN has been trained to map input vectors to their corresponding output vectors. The training is an iterative process where at each iteration; another input data is randomly selected from the same training dataset to become the input signals for the networks. The process continues again until certain condition of MSE value is respected. The lower the value of MSE, the more identical input-output nodes could have been obtained. In our case, the training was stopped when the MSE was lower than 0.001.

Once the training is completed, the final weights obtained in the networks are fixed. This trained-AANN is then separated into two different networks: compression and decompression networks, as previously shown in Figure 2. The bottleneck layer of AANN becomes its output layer in this trained-AANN-compression network. With this structure, AANN compression network can be considered as a dimension reduction model from high dimensional data into low dimensional data as shown in Figure 3.

During testing stage, all high-dimensional input data from testing dataset are inserted to the trained-AANN-compression networks. For each of the data, the values at bottleneck nodes are calculated using the final weights previously obtained during AANN training stage. The reduced dimension of high-dimensional data is represented by the values at bottleneck nodes. These values can be clustered on one or two-dimensional plane in order to visualize the similarities and differences between these data.

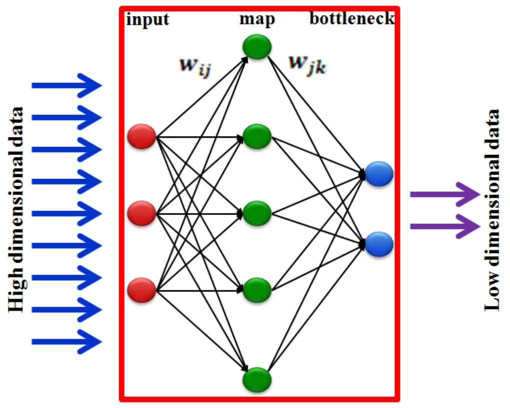

Figure 3: The trained-AANN-compression networks. The final weights are obtained after the training of the whole AANN is completed.

On the other hand, it can also be noticed that oppositely, the trained-AANN-decompression networks should have the ability to reverse dimension reduction process as its structure consists of bottleneck layer as input layer and its output is also AANN's output layer. It could then conceptually decompress low dimensionality data into higher dimensionality data at its output layer. (This feature of AANNs however was not given any consideration in this paper).

2.2 Evaluation of AANN Clustering Performance

Using AANN, depending on the size of the bottleneck, the clustered data can usually be visualized on one or two-dimensional plane. However, it is quite difficult to measure the performance of the clustered data whether they are appropriately classified or misclassified according to their respective classes by using only visualization. In this research, a method to comparatively measure the accuracy of the data clustering has been used. This method is based on the concept of closeness of data to its known class. The Iris flowers and Italian olive oils datasets have been divided into two parts: training and testing sets. Initially, all the final weights obtained during training were saved. These final weights have subsequently been used to establish the AANN compression and then cluster the whole input dataset one by one on one or two dimensional map as shown in Figure 4.

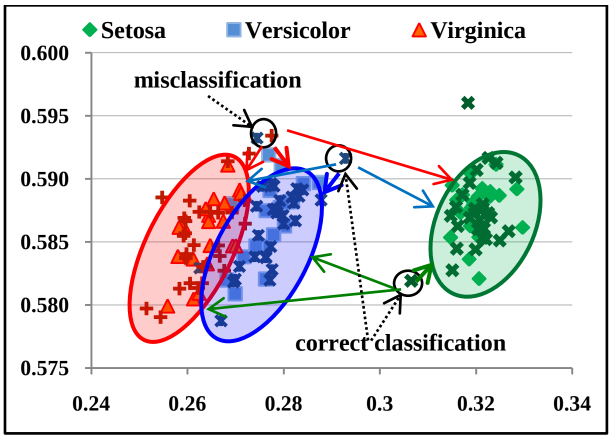

Figure 4: AANN data clustering accuracy. One test data is considered as misclassified while others are correctly classified

In principle, a particular class of testing data should have been clustered closer to its own class rather than other classes. In our work, a correct classification, *C(x)*, of a clustered test data is measured using the following equation:

$$C(x) = \begin{cases} 1 & \text{if average distance to its own class is minimum} \\ 0 & \text{otherwise} \end{cases}$$

(eq-9)

Therefore, the calculation of AANN data clustering accuracy has been defined as:

$$Accuracy(\%) = \frac{Number of Correct classification}{Number of Test data} * 100$$

(eq-10)

where a correct classification represents a test data that has been correctly clustered closer to its class while misclassification represents otherwise. Using this data accuracy calculation, the higher accuracy indicates that higher number of test data have been clustered closer to their appropriate and respective classes.

2.3 AANN Clustering Performance

Two types of experimental works have been conducted with the purpose to investigate the AANN's clustering performance and ability in performing the tasks of dimension reduction, data clustering and data visualization using various sizes of training and testing datasets. Two different datasets have been used: Iris flowers and Italian olive oils datasets. The Iris flower dataset has 150 samples with 4 attributes while the olive oil dataset has 572 samples with 9 attributes. The structure of the AANN used had five layers as illustrated in Figure 2. According to the study on NLPCA using AANN [1], this structure was essential to achieve optimal non-linear feature extraction. Two bottleneck nodes have been used, similar to [2], all input data were normalized into the range of 0 and 1. In addition to that, the data clustering accuracy as described in 2.2 has also been applied to measure the performance of AANN in clustering high-dimensional data. Both Iris flowers and Italian olive oils datasets have been randomly divided into two sub datasets: training and testing datasets as shown in Table 1.

Table 1: Number of data randomly selected in training and testing datasets of Iris flowers and Italian olive oils datasets

No	% training	Iris flowers		Olive oils	
		Train	Test	Train	Test
1	93.3	140	10	533	39
2	80.0	120	30	457	115
3	66.7	100	50	381	191
4	53.3	80	70	305	267
5	40.0	60	90	228	344
6	26.7	40	110	152	420
7	13.3	20	130	76	496
8	6.7	10	140	38	534

In Table 1, eight experiments have been conducted on each Iris flowers and Italian olive oils datasets. The ratio of total number of data presented in training and testing datasets varied from 93.3% (training):6.7% (testing) to 6.7% (training):93.3% (testing). In Iris flowers dataset, it varied from 140:10 to 10:140 while it was from 534:38 to 38:534 in Italian olive oils dataset. For each type of these experimental works, the program was run fifteen times and the average values were taken. All experiments were conducted using Intel Core2Quad 2.5 GHz with 2.0 Gb memory.

3. CASE STUDIES AND RESULTS

3.1 Case Study-1: Iris Flowers Data

In CaseStudy-1, Iris flowers dataset has 150 samples (flowers) and they are equally divided into three different classes: Setosa, Virginica and Versicolor. Each sample has four attributes measured in cm: length (sepal), width (sepal), length (petal) and width (petal). The full dataset as shown in Table 2 can be found in [16]. The structure of AANN used consisted of four nodes at input and output layers, ten nodes at map and de-map layers, and two nodes at bottleneck layer (x and y coordinates). At first, the whole dataset has been used as training data set for AANN. Once the network is trained, the data are then clustered by the network onto two-dimensional plane as shown in Figure 5.

Table 2: Input data matrix of 150 Iris species. Each of them has four attributes namely Sepal Length, Sepal Width, Petal Length and Petal Width.

Attributes

		Sepal-L	Sepal-W	Petal-L	Petal-W	Species
Species/Item	I_1	5.1	3.5	1.4	0.2	se
	I_2	4.9	3.0	1.4	0.2	se
	...	...	...	...	...	...
	I_{150}	5.9	3.0	5.1	1.8	vg

*se = setosa, vc = versicolor, vg = virginica

In Figure 5, it can be observed that Setosa and Virginica flowers have been well clustered far from each other, while the class of Versicolor flower has been clustered between these two classes, and closer to Virginica class. This distribution of Iris flowers species was consistent with the natural classification of the Iris flowers dataset itself as described in [16].

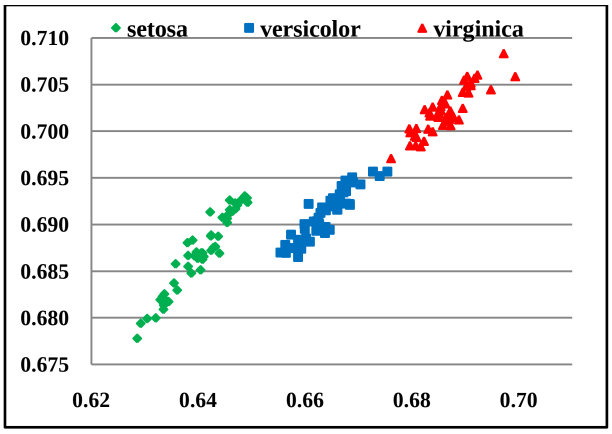

Figure 5: The clustering of whole Iris flowers dataset defined by the two activation nodes at bottleneck layer in AANN.

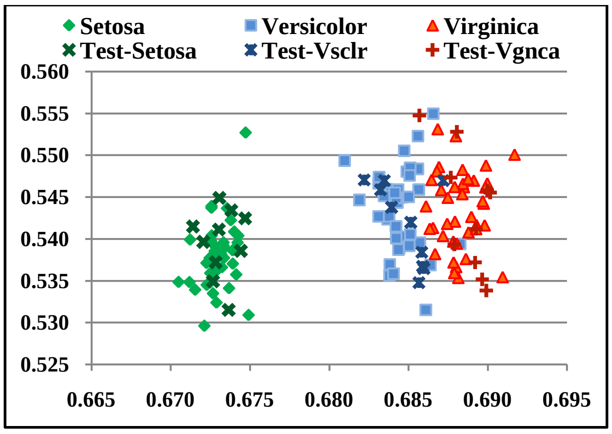

Figure 6: The AANN clustering of 120 training data and 30 testing data of Iris flowers. Large numbers of test data are well clustered closer to its own classes.

Different proportions of randomly selected training and testing datasets of Iris flowers are shown in Table 1. These datasets have been used for the training and testing of AANN and its performance evaluation according to eq-9 and eq-10.

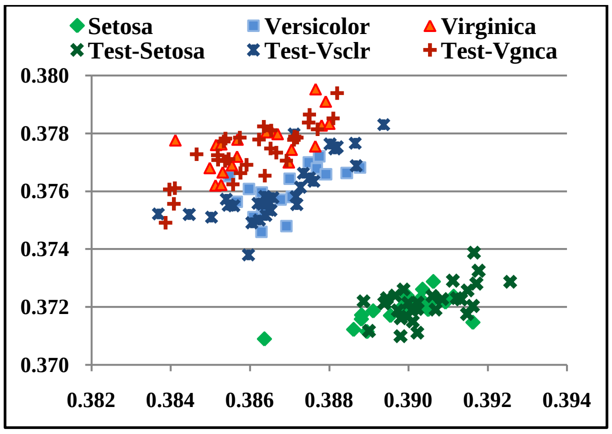

Figure 7: The AANN clustering of 60 training data and 90 testing data of Iris flowers. Large numbers of test data are well clustered closer to its own classes even though the number of training data is smaller than the number of testing data.

The test datasets were clustered using AANN previously trained by the training datasets. The examples of Iris flowers data clustering using 120:30 and 60:90 training:testing datasets are shown in Figure 6 and Figure 7 respectively. In these figures, it can be observed that most of the test data have been clustered closer to their own classes. This indicates that the trained AANN has been able to cluster appropriately very large number of test data closer to their own classes. It can also be seen (in Figure 7) that AANN has been able to cluster majority of the test data closer to their respective classes even though the network was trained with smaller proportion of training dataset. The AANN clustering performance (Section 2.2) was used to measure the degree of closeness of the test datasets with respect to their known classes. In this work, our data clustering accuracy rate has been measured using AANN with one and two bottleneck nodes as shown in Figure 8.

In Figure 8, the data clustering accuracy performance of AANN has been consistently above 93.3% even though the number of samples in training dataset decreased from 140 to 10. This means that the AANN has been able to learn the inherent characteristics of Iris flowers dataset using only small proportion of the total dataset for its training. Expectedly, the best accuracy rate obtained by AANN was 98.9% when it has been trained and tested by 140 and 10 data respectively and using two bottleneck nodes. The range of accuracy of data clustering using AANN with one bottleneck was 94.2% to 97.3% while for the case of AANN with two-bottleneck nodes; it was from 93.3% to 98.9%. For Iris flowers clustering, there was not a significant different between the average performances of AANN with one or two bottleneck nodes.

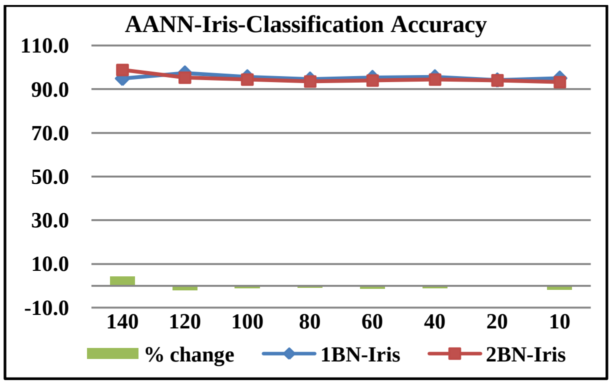

Figure 8: The performance of data clustering accuracy of AANN in clustering Iris flowers dataset according to the number of training and testing data used. % indicates the different performance of AANN with two bottleneck nodes compared to AANN with one bottleneck node.

3.2 Case Study-2: Italian Olive Oils Data

The second dataset that has been used in the Case Study-2 was Italian olive oils dataset. As described in [2], this dataset contains the concentrations of eight fatty acids namely palmitic, palmitoleic, stearic, oleic, linoleic, eicosanoic, linolenic, eicosenoic. Concentrations of these fatty acids were measured for 572 samples of olive oils in Italy. The data was taken from nine different growing regions in Italy, which are North Apulia, Calabria, South Apulia, Sicily, Inland Sardinia, Coastal Sardinia, East Liguria, West Liguria and Umbria. Figure 9 shows the geographical locations of these samples taken from nine different regions in Italy.

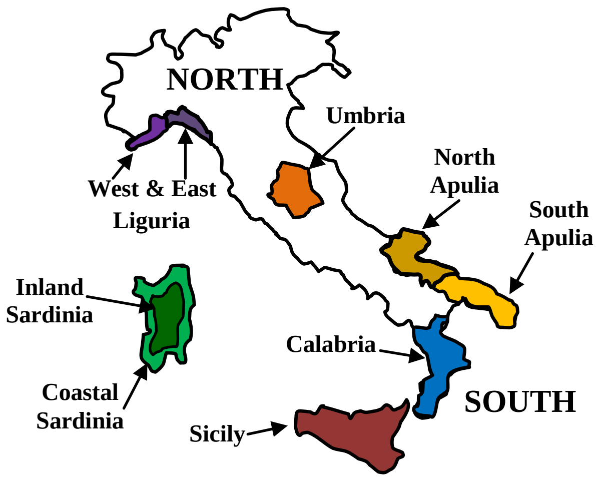

Figure 9: The geographical locations of Italian olive oils samples.

In this case study, the structure of AANN had eight nodes at input and output layers, twenty nodes at map and de-map layers, and two nodes at bottleneck layer. As in the previous case study, the high dimensional data of Italian olive oils dataset has also been reduced to one and two dimensional data represented by the number of neurons in bottleneck nodes. At the first step, the whole dataset has been used as training data set for AANN and subsequently used for clustering as shown in Figure 10.

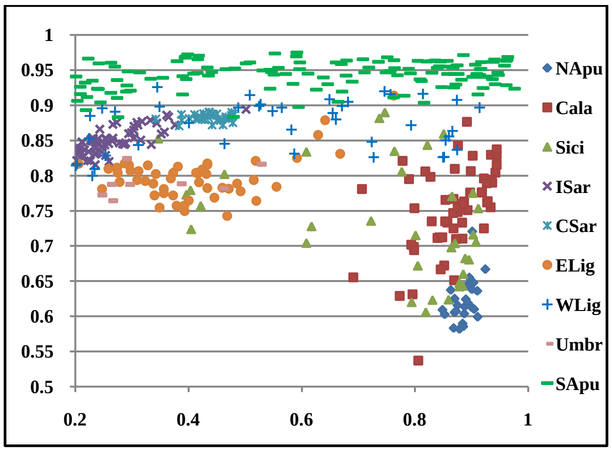

Figure 10: The clustering of Italian olive oils dataset defined by the two activation nodes at bottleneck layer in AANN.

Figure 10 shows that samples taken from four South-East regions of Calabria, Sicily and North Apulia have formed a group on the right side. On the other hand, another group was formed on the left side that contained samples taken from the three North-West regions of Umbria, West Liguria, East Liguria and island of Sardinia (Inland and Coastal). Meanwhile, the largest samples that were taken from South Apulia were clustered at the top from left to right side above the two groups. Similarly as in [2], these samples were far from North Apulia and were closer to Inland and Coastal Sardinia. In the left group, these Inland and Coastal Sardinia samples were closer to each other and have formed a subgroup which were different from East Liguria, West Liguria and Umbria.

The clustering performance of AANN with one and two bottlenecks is shown in Figure 11. It can be seen that the performance of AANN with two bottleneck nodes is better than AANN with one bottleneck node in clustering Italian olive oils dataset.

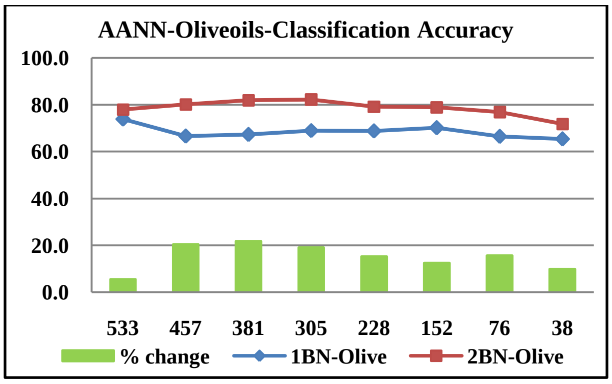

Figure 11:The performance of data clustering accuracy of AANN in clustering Italian olive oils dataset according to the number of training and testing data used.% indicates the different performance of AANN with two bottleneck nodes compared to AANN with one bottleneck node.

In this case-study we also used various proportions of training and testing datasets as presented in Table 1. Figure 11 shows that data clustering accuracy performances for both cases of one and two bottlenecks have slightly decreased when the proportion of data used for training is decreased. The best rate achieved was 82.1% by AANN with two bottleneck nodes using 305 training data and 267 testing data. The range of AANN data clustering accuracy using one bottleneck node was 66.4% to 73.8% while for the case of AANN with two-bottleneck nodes; it was 71.8% to 82.1%. Through all these experiments of clustering Italian olive oils dataset, the average performance of AANN with two bottleneck nodes has been about 15% higher than that of AANN with one bottleneck node.

4. DISCUSSION

From the experimental results of clustering Iris flowers and Italian olive oils datasets, our developed AANN algorithm has been able to cluster large number of testing data into their known classes. From these results too, the grouping of the data clusters was found agreeable to the known and inherent characteristics of the datasets themselves. Iris data seems to be better clustered into its respective classes compared to Italian olive oils data. This is certainly influenced by fewer clusters and dimensions of Iris data. The data clustering performance equation has been able to measure the performance of the clustered data in terms of their closeness (correctness) of classification. Through this investigation, the AANN with one or two bottleneck nodes achieved data clustering accuracy of more than 93.3% for Iris flowers dataset with the best rate of 98.9%. However, its performance was lower in clustering Italian olive oils where the best rate achieved was 82.1%. AANNs with one or two bottlenecks have been able to demonstrate that their ability to nonlinearly reduce the dimension of Iris and olive oil datasets, but did not remove inherent characteristics of each dataset allowing them to be classified with high levels of accuracy. It can also be concluded that it is going to be a bigger challenge for AANN to cluster datasets with very high dimensionality such as gene expression dataset that could have hundreds or thousands features. AANNs with two bottleneck nodes have been able to perform better when compared to AANNs with only one bottleneck node. With additional number of bottleneck nodes, AANNs should be able to improve its clustering and data compression ability whilst retaining most of information that has been compressed as shown in Italian olive oils clustering. The more complexity in a dataset such as higher number of samples, more similar samples, higher features or dimensionality of samples and also higher number of classes could all contribute to the difficulty of having good performance of data clustering using AANNs.

5. CONCLUSION

In this paper, the technique of AANN has been discussed and developed using high level language of C/C++, based on the structure, equations and rules described in Section 2. Our developed AANN algorithm has been used to perform compression, clustering and visualization of high-dimensional data using two datasets: Iris flowers and Italian olive oils. The technique of AANN data clustering accuracy has been successful to measure the data clustering accuracy performance of this algorithm. The experimental results have shown that AANN algorithm has been able to cluster these two high-dimensional data according to their respective classes. Finally, with this ability, it could give us an alternative method to explore the potentiality to enhance multidimensional data clustering technique by integrating it with another technique such as the Self Organizing Maps with the purpose to cluster very complex multidimensional data such as gene expression data with multidimensional variability as described in [17].

6. ACKNOLEDGMENTS

This study was supported by Ministry of Higher Education Malaysia and Universiti Teknologi Malaysia (UTM) through research grant GUP04H40. Thanks are also due to Newcastle University Upon Tyne for providing assistance and facilities in this study.

7. REFERENCES

[1] M. A. Kramer, "Nonlinear Principle Component Analysis Using Autoassociative Neural Networks," AIChE Journal, vol. 37, pp. 233-243, 1991.

[2] B. W. D. L. M. M. Daszykowski, "A Journey Into Low-dimensional Spaces With Autoassociative Neural Networks," International Journal of Pure and Applied Analytical Chemistry Talanta, vol. 59, pp. 1095-1105, 2003.

[3] L. I. Smith, 2002. [Online]. Available: http://neurobot.bio.auth.gr/2005/a-tutorial-on-principal-components-analysis/. [Accessed 05 February 2012].

[4] S. L. J. L. Giraudel, "A Comparison of Self Organizing Map Algorithm and Some Conventional Statistical Methods for Ecological Community Ordination," Journal of Ecological Modelling, vol. 146, pp. 329-339, 2001.

[5] F. V. N. T. M. Jaisheel Mistry, "Missing Data Estimation Using Principle Component Analysis and Autoassociative Neural Networks," Journal of Systemics, Cybernatics and Informatics, vol. 7, no. 3, pp. 72-79, 2009.

[6] V. M. Stone, "The Autoassociative Neural Network-A Network Worth Considering," in World Automation Congress (WAC), Hawaii, 2008.

[7] P. J. K. M. G. Ali Anaissi, "A Framework for High Dimensional Data Reduction in the Microarray Domain," in IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010.

[8] T. Kohonen, "The Self-Organizing Maps," in Proceedings of the IEEE, 1990.

[9] A. Flexer, "On the Use of Self Organizing Maps for Clustering and Visualization," in International Conference on Principle on Data Mining and Knowledge Discovery, Prague, Czech Republic, 1999.

[10] E. A. Juha Vesanto, "Clustering of the Self Organizing Maps," IEEE Transaction on Neural Networks, vol. 11, no. 3, pp. 586-600, 2000.

[11] m. F. K. C. L. G. J. K. Matthias Scholz, "Non-linear PCA: A Missing Data Approach," Journal of Bioinformatics, vol. 21, no. 20, pp. 3887-3895, 2005.

[12] S. Y. B. C. M. A. Abidi, "Image Compression Using Hybrid Neural Networks Combining The Auto-Associative Multilayer Perceptron and The Self Organizing Feature Map," IEEE Transaction on Consumer Electronics, vol. 40, no. 4, pp. 796-811, 1994.

[13] B. J. H. J. D. L. Mark J. Embrechts, "Augmented Efficient BackProp for Backpropagation Learning in Deep Autoassociative Neural Networks," in International Joint Conference on Neural Networks (IJCNN), Barcelona, 2010.

[14] J.-C. G. Gaetan Kerschen, "Feature Extraction Using Auto-Associative Neural Networks," Smart Materials and Structures, vol. 13, no. 1, 2004.

[15] M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems 2nd Edition, Pearson Education Limited, 2005.

[16] R. A. Fisher, "UCI Machine Learning Repository," 2006. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Iris. [Accessed 31 January 2011].

[17] E. Mesbahi, "Cryptic codes in non-coding DNA: Autoassociative Neural Networks and multidimensional Self Organising Maps (SOM) mediated prediction of positional significance of cis-elements in co-regulated expression systems," 29 March 2012. [Online]. Available: http://www.ncl.ac.uk/marine/research/project/1997. [Accessed 15 July 2012].